A HMM-Based Generative Model for Gesture Recognition

Francisco Cai, David Philipson, Nikil Viswananthan

March 16, 2010

1 Introduction

In our project, we apply probabilistic graphical models to the task of simple gesture recognition. Even a simple gesture would vary a fair amount from individual to individual and thus it would be difficult to devise a formal definition for a particular gesture. However, a probabilistic model seems well-suited to capture those variations naturally. Specifically, we are given the 3-D joint locations of individuals while performing nine simple gestures. From this data, we learn a hidden Markov model (HMM) for each of the gestures, and use these models to classify unseen gestures.

In the next section, we will discuss our motivations for using a HMM, and related work that have applied HMMs to gesture recognition. In the following two sections, we will describe our methods and implementation in more detail, and in the last two sections, we analyze our results and present the conclusions we have drawn from this project.

2 Motivations and Related Work

There are a few reasons that make HMMs attractive for the task at hand. First, gestures are intrinsically temporal and HMMs are designed to model observations occurring over time. Second, gestures can be separated into distinct phases based on their movement of the joints – this naturally leads to an HMM model where the phases of the gesture are the hidden states and the joint movements corresponding to a phase are the emissions of those hidden states. Third, HMMs are relatively simple models and our first priority is to have a simple system working.

The literature shows that using HMMs is indeed a reasonable strategy. We found in the literature that HMMs have been successfully applied to gesture recognition with work done as early as 1994, by Jie Yang and Yangsheng Xu at Carnegie Mellon University. In their paper "Hidden Markov Model for Gesture Recognition", they report a 99.78% accuracy when recognizing nine different isolated gestures . Others have extended the use of HMMs to recognize up to 20 gestures with more than 90% accuracy (Yang, 94) and recognize words in American Sign Language in real-time (Starner, 95).

Our project differs from the aforementioned work in two ways: first, we are working with full-body gestures with depth information, whereas the work done by Yang and Xu only recognized gestures that corresponded to writing out the digits 1 through 9 in two dimensions. We are also working with simpler features – joint movements – whereas the authors of the above papers made use of Fourier transforms. Again, we focus on simple features for faster prototyping, but as it turns out we are able to obtain very high classification accuracy with just these features.

3 Methods and Extensions

3.1 Data

3.1.1 Collection

Our dataset was provided by Hendrik Dahlkamp, a PhD student in the Stanford Computer Science Department. Using a Kinect sensor, he and other members of his lab recorded themselves doing nine gestures – clap, jump, flick left, flick right, high kick, low kick, punch, throw, and wave. This data was then passed to a skeleton tracker which they had previously developed. This tracker converted the raw Kinect sensor data into the 3D positions of twenty joints, such as elbows, hips, and wrists, for each frame in the recording. Our dataset consisted of 121 gesture files, each containing the 3D positions of joints throughout a single gesture.

3.1.2 Visualization and Data Cleaning

In order to visualize the data, we built a tool to read in the frames, model the three dimensional skeleton, and provide controls for stepping through the frame animations. When testing the visualization system, we

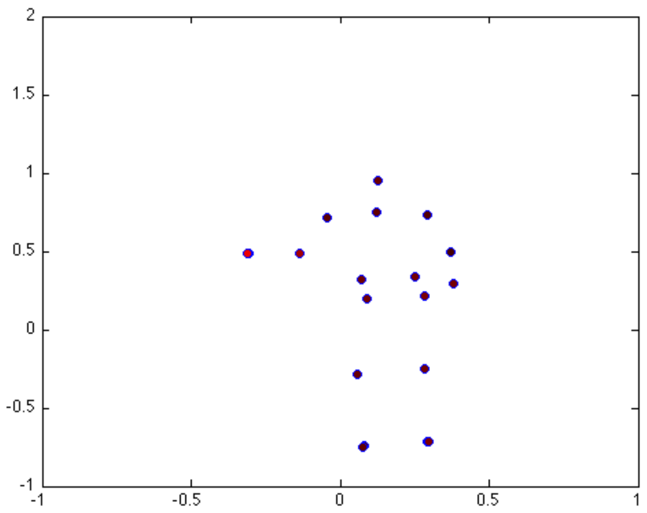

Figure 1: Visualization of one frame in the middle of a "left flick" sequence.

discovered that the training gesture files contained anomalies that could adversely affect our learning:

- Extraneous motion before and after the gesture;
- Gestures files labelled as containing one gesture that actually contained several gestures in series.
- Sequences of the creator walking to the laptop that was connected to the Kinect were inadvertently saved as gesture files.

We built additional tools to clean up our dataset. However we did not see a performance increase after cleaning the data.

3.2 HMM Model

3.2.1 Intuition and Description

The intuition behind our model for gestures is that a gesture consists of distinct phases where the joint velocities and joint angular velocities are similar. For a simple example, a jump gesture can be thought of having up and down phases, where the former is characterized by positive velocities in the y direction and the latter is characterized by negative velocities in the y direction.
With this intuition in mind, we model each gesture with an HMM, where:

1. The states of the hidden variable correspond to the different phases of the gesture. From our intuition of the gestures, we believed that having three states would be a good starting choice.
2. Given a state of the hidden variable, the emission is a vector of joint velocities drawn from a multivariate Gaussian distribution.
3. The transition probabilities are such that, from any given state, we may transition to the same state (with high probability), or to the next state (with low probability), but we may not transition backwards or skip a state.

4. The parameters, the transition and emission probabilities, are learned from the training examples.

3.2.2 Learning

To learn the parameters for the HMM for a particular gesture, we first collect all the files containing 3D joint positions of the joints for this gesture. For each file, we convert the sequence of joint positions into joint displacements from the previous time step, which we can treat as a sequence of emissions from the HMM. Using these sequences of emissions as training data and specifying a number of hidden states, we can then use expectation-maximization to learn the best-fit parameters for our HMM.

3.2.3 Testing

We decided to test primarily with leave-one-out cross-validation, given that our dataset did not have many examples to begin with. When we received more training data, we ran some tests using leave-two-out cross-validation. We did not run any tests with a larger test set / training set split – the "right flick" gesture had only twelve examples, even after receiving the new data, so leaving more than 2 examples out at a time would be substantial portion of our dataset for that gesture.

4 Implementation

4.1 Model Construction

To implement the HMMs needed for our model, we turned to the PMTK toolkit, a library providing MATLAB functions for a variety of graphical models. In order to train HMMs from our recorded data, we specify a number of hidden states, then use the PMTK toolkit to run EM in order estimate the parameters: the transition probabilities between the hidden states and the parameters of the CPDs (that is, the mean vector and covariance matrix for each state).

By running the default version of EM, we would have no guarantee that the transition probabilities would have the desired property that from any given state, the only possible transitions are to the same state or the next one. We tried two approaches to implement this. First, we set the initial transition matrix to have zeros in all entries which we wished to disallow. This seemed to encourage the desired probabilities to be high, but did not enforce that the undesired probabilities be zero. If we desire to force zeros, we can also set the prior counts to be all zero, setting the goal to MLE rather than MAP from a Dirichlet prior.

We found the results to be extremely similar with both versions, but the MLE version took longer to compute. Hence, in our final algorithm we opted to use the default MAP estimates.

4.2 Classification

Our initial implementation was, given an emission sequence from an unlabelled gesture, to simply compute the (log) likelihood that the HMM for each gesture could have generated it, and then to choose the label from whichever machine gave rise to the greatest likelihood. We experimented with a number of other more elaborate methods for classification, but none proved more effective than this naive implementation. Our attempts and their results are discussed in the next section.

5 Results and Analysis

5.1 Initial Results

At our initial run of the program, with no features other than the 3-dimensional positions of the joints and no special inputs to the algorithm, we obtained an accuracy of 75.9 percent, which suggested to us that our approach was promising. Noticing that many of our data samples had anomalies such as a mixture of several gestures or long periods of walking or standing before or after the gesture, we elected to attempt to clean the data samples to distill them to just the gestures we wanted to learn, as we believed that these anomalies in the data samples might be obscuring the effectiveness of our implementation.

5.2 Data Cleanup

We cleaned out these anomalies in our training examples and reran our earlier performance evaluations with leave-one-out cross-validation, expecting better results. However, we were surprised to find that our overall accuracy actually decreased all the way down to 63.0 percent.

Our confusion matrix revealed some patterns (see end of report for our data), and allowed us to guess at why this happened. With the new results, our algorithm rarely classified a test example as certain gestures (in particular, it did not classify any gesture as a right flick), while on the other hand certain gestures were guessed overly often (for example, 40 percent of the missed predictions classified as jump). We therefore believe that the decrease in accuracy after cleaning the examples occurred because the types of the gesture are now more distinct. Hence, the HMMs associated with the different gestures are more distinct from each other, and in particular some of them tend to always output likelihoods which are higher than the likelihoods output by other HMMs. In other words, now that the HMMs were more dissimilar, we could no longer expect their baseline likelihood predictions to be close to each other.

5.3 Additional Features

The first step we decided to take was to include some additional features. In addition to the 3-dimensional coordinates, we thought it might be useful to add certain joint angles as features. For example, explicitly stating the elbow angle as a feature would likely be useful for identifying a throwing motion, rather than leaving Gaussian parameter estimation to infer the relationship between the positions of the arm joints. We settled on a few joint angles that we believed most useful for identifying pose: elbow, knee, and hip joints. This addition saw a very modest improvement in accuracy, from 63 to 67 percent.

5.4 Initial Distributions

Our next addition was to specify initial distributions on the state transitions, as discussed in the Implementation section. By simply giving an initial distribution to encourage the current-or-next-state transitions of our envisioned model, we saw another small increase in accuracy, this time to 71 percent. As discussed above, we also tried MLE instead of MAP estimates, but these took longer to compute and saw virtually no change in the results. We concluded that although the MAP values may not have strictly zero values in the transition probabilities we would like, they are close enough that our eventual classification remains essentially the same.

5.5 More Data

Our biggest improvement came after noticing that the classification which consistently performed worst, right flick, was also the one with the fewest training examples. Compared to the other gestures which had from 12 to 20 examples, right flick had a mere eight, which was reduced to seven for training when running LOOCV. Thinking this might be a problem, we requested more samples from Hendriks lab. We thus obtained enough new samples to roughly double the size of our dataset. By simply including these additional training examples, our overall accuracy skyrocketed from 71 to 92 percent. We thus conclude that the success of our algorithm can be greatly improved by increasing the size of the training set.

5.6 HMM "Normalization"

We now set out to resolve the problem of some gestures which receive too many classifications, and others which receive not enough. As discussed above, we believe that this may be caused by the HMMs associated with some gestures tending to always return higher or lower likelihoods. Hence, we sought a way to normalize the HMMs, by finding a baseline likelihood for each. We could then compare the output likelihoods given a test example to an HMMs baseline, rather than using absolute likelihood to make our classifications.

We attempted to find an HMMs baseline by examining its likelihood of producing each negative example (that is, each example in the training set from a gesture other than its own). Presumably, if an HMM outputs a likelihood well-above its expected likelihood on negative examples, then it is a good indication that the given example is its gesture type. We thus took each HMMs baseline to be the average of its (log) likelihood over all negative training examples. Unfortunately, this drastically damaged our accuracy, taking it all the way down to 23 percent. While the originally over-classified gestures were no longer such, instead they were now under-classified, and vice versa. This suggests that our baseline adjustment to likelihood was too strong, so that we overshot the correct normalization. We attempted several other variants, such as measuring the number of standard deviations that the observed likelihood was away from the average negative likelihood, but none of the methods we tried led to greater accuracy than our 92 percent without any baseline adjustment.

If we had more data, we would have liked to set aside a validation set containing positive examples to estimate the baseline taking into account the (log) likelihoods of the positive examples as well.

5.7 Number of States

We tried running our algorithm specifying different numbers of hidden states. We were surprised to find identical performance for any of one, two, or three hidden states. Four hidden states had slightly worse performence, at 89 percent accuracy, and any additional states led to lower performance.

5.8 Feature Selection

Our last attempt to improve accuracy was to try to select features for each machine. We thought that perhaps only some of the features were relevant to a given gesture, and by building that gestures machine around those features we could more accurately classify that gesture. To select features, we first generated HMMs using all features, in the manner we had been using in all previous tests. Then, for each HMM we looked at the parameters for the Gaussians in the CPDs for each state. If a given emission element had mean with high absolute value and low standard deviation, then we consider it more important, since it represents more certainty about motion in that particular feature. Assigning an importance score to each feature based on these metrics, we then select a chosen number of most important features and retrain the HMM using just those features. Unfortunately, every method we tried of assigning importance scores caused drastic reductions in accuracy for any significantly smaller number of features; our best accuracy with any such score was a mere 30 percent, with the problem of over-classified and under-classified gestures greatly exacerbated. We therefore believe that the accuracy drop was caused by reasons similar to those discussed above when we cleaned the data; by making the HMMs more distinct (here, different machines could train on entirely different features), we make it likely that different machines will have very different baseline likelihoods. We hence decided to move away from feature selection as a component of our algorithm.

5.9 Results Summary

This table summarizes our best classification results, from training on the larger, final dataset. For these results, we kept the adjustments (described above) that were helpful – we cleaned up the data and specified the initial distributions. Our overall accuracy was 93.5185 % on this run. Generally, our accuracy was above 91%, due to the random initializations in the EM algorithm when learning parameters.

6 Conclusion

Our high accuracy with basic features and customization suggest that HMM-based models show great promise for gesture recognition. HMMs have the advantage of running quickly and of interfacing easily with existing software libraries. We can see a number of directions for further study.

Gestures:	Flick Left	Flick Right	High Kick	Jump	Low Kick
Accuracy (%)	91.67	100.00	100.00	100.00	83.33
Precision	1.000	1.000	0.857	0.7059	1.000
Recall	0.917	1.000	1.000	1.00	0.833
Gestures:	Punch	Throw	Wave	Clap	
Accuracy (%)	83.33	100.00	83.33	100.00	
Precision	1.000	1.000	1.000	1.000	
Recall	0.833	1.000	0.833	1.000	

First, even at the end with our 93% accuracy, most of our errors occurred due to guessing certain gestures too readily while others not readily enough. While we could not find an effective "normalization" procedure to even out the likelihoods produced by the different machines, we believe that such a method is most likely possible. As discussed above, we believe we could make good use of a larger dataset to compare the output likelihoods on positive and negative examples.

Another line of study would be to take advantage of the nature of HMMs, which evaluate the likelihood of a sequence "up to the present," to investigate online gesture recognition, i.e. attempting to recognize gestures while they are in progress.

One result that surprised us was the overall robustness of the algorithm. Most of the changes we tried produced accuracies that varied by less than ten percent, while still remaining at high accuracy. Even fairly significant adjustments like greatly modified feature sets, hard restrictions on the machines' transition probabilities, or different numbers of states caused relatively small changes in accuracy, for greater or for worse. We found that most of our improvements were incremental, each earning a few extra points of accuracy. Given our current high rate, it seems likely that further improvements will be of this nature as well.

7 Data

With our final version, our confusion matrix appears as follows, where the rows correspond to our algorithm's classifications, while the columns correspond to the correct classifications.

The gestures are, in order: left flick, right flick, high kick, jump, low kick, punch, throw, wave, and clap.

$$\begin{pmatrix}
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 12 & 0 & 2 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 12 & 0 & 2 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12
\end{pmatrix}$$

Observe that every incorrect prediction was classified as either the third or fourth gesture, respectively "jump" and "high kick."

8 Bibliography and Acknowledgements

Chen, F.S., Fu, C.M. and Huang, C.L. *Hand gesture recognition using a real-time tracking method and hidden Markov models*. Image and Vision Computing, 2003, Vol 21-8, 745-758.

Starner, T. and Pentland, A. *Real-time american sign language recognition from video using hidden markov models*. International Symposium on Computer Vision, 1995. Proceedings., 265–270.

Yang, J. and Xu, Y. *Hidden markov model for gesture recognition*. Robotics Institute, Carnegie-Mellon University, 1994.

Dunham, M., Murphy, K and various authors. *Probablistic Modelling Tookit for Matlab/Octave (PMTK3)*. Software Package, MIT License, 28-Feb-2011.

We would like to thank Hendrik and his colleagues for recording gesture data for us and for their advice and guidances; Daphne Koller for the modelling advice; and the authors of PMTK3, the Matlab package we used for our project.